**Android TouchEvent事件传递机制**

1、跟touch事件相关的3个方法：

public boolean dispatchTouchEvent(MotionEvent ev); //用来分派event

public boolean onInterceptTouchEvent(MotionEvent ev); //用来拦截event

public boolean onTouchEvent(MotionEvent ev); //用来处理event

拥有这三个方法的类：

Activity类： Activity dispatchTouchEvent();

onTouchEvent();

View容器（ViewGroup的子类）：

FrameLayout、LinearLayout……

ListView、ScrollVIew……

dispatchTouchEvent();

onInterceptTouchEvent();

onTouchEvent();

View控件（非ViewGroup子类）：

Button、TextView、EditText……

dispatchTouchEvent();

onTouchEvent();

2、三个方法的用法：

（1）**dispatchTouchEvent() 用来分派事件。**

其中调用了onInterceptTouchEvent()和onTouchEvent()，一般不重写该方法

（2）**onInterceptTouchEvent() 用来拦截事件。**

ViewGroup类中的源码实现就是{return false;}表示不拦截该事件，事件将向下传递（传递给其子View）；

若手动重写该方法，使其返回true则表示拦截，事件将终止向下传递，事件由当前ViewGroup类来处理，就是调用该类的onTouchEvent()方法

（3）**onTouchEvent() 用来处理事件。**

返回true则表示该View能处理该事件，事件将终止向上传递（传递给其父View）；

返回false表示不能处理，则把事件传递给其父View的onTouchEvent()方法来处理。

3、（1）这一系列的传递流程都是dispatchTouchEvent()方法来控制的，如果不人为地干预，事件将由上自下依次传递（因为默认是返回false不会拦截的），传递到最底层的View，就由它的onTouchEvent()方法来处理事件，若处理成功返回true，若处理失败返回false，事件依次向上传递，每个View都调用自己的onTouchEvent()方法来处理事件，若处理成功就终止传递，若处理失败就继续向上传递。

（2）经过人为的干预，若在向下传递的过程中被拦截了，即**onInterceptTouchEvent()方法返回true，则事件将停止向下传递，直接由当前的onTouchEvent()方法来处理，若处理成功则OK，若处理不成功，则事件会向上传递。**

（3）**另外，dispatchTouchEvent()方法中还有“记忆”的功能，**如果第一次事件向下传递到某View，它把事件继续传递交给它的子View，它会记录该事件是否被它下面的View给处理成功了，（怎么能知道呢？如果该事件会再次被向上传递到我这里来由我的onTouchEvent()来处理，那就说明下面的View都没能成功处理该事件）；当第二次事件向下传递到该View，该View的dispatchTouchEvent()方法机会判断，若上次的事件由下面的view成功处理了，那么这次的事件就继续交给下面的来处理，若上次的事件没有被下面的处理成功，那么这次的事件就不会向下传递了，该View直接调用自己的onTouchEvent()方法来处理该事件。

（4）“记忆”功能的信息只在一系列事件完成之前有效，如从ACTION\_DOWN事件开始，直到后续事件ACTION\_MOVE，ACTION\_UP结束后，“记忆”的信息就会清除。也就是说如果某View处理ACTION\_DOWN事件失败了（onTouchEvent()返回false），那么后续的ACTION\_MOVE,ACTION\_UP等事件就不会再传递到该View了，由其父View自己来处理。在下一次发生ACTION\_DOWN事件的时候，还是会传递到该View的。

1、Touch事件的相关细节（发生触摸的位置、时间等）被封装成MotionEvent对象。

先简单介绍下这个TouchEvent是什么？安卓手机的交互，主要就是手指在屏幕上的戳戳滑滑点点，而我们的这些操作其实主要是由三种基本动作组成的：按下down、移动move、抬起up，安卓中把这个基础动作叫做TouchEvent。

比如：一次点击就是按下、抬起组成的；一次长按就是按下、等待、抬起组成；一次滑动操作则是，按下、移动、抬起组成。

事件类型 具体动作

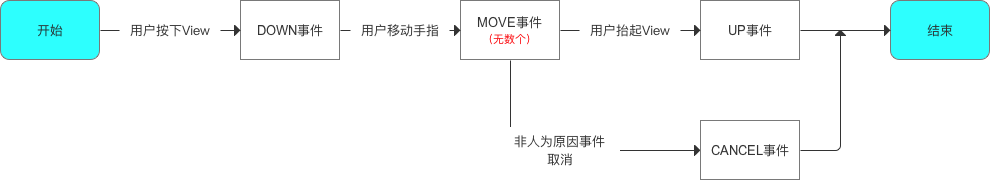
MotionEvent.ACTION\_DOWN 按下View（所有事件的开始）

MotionEvent.ACTION\_UP 抬起View（与DOWN对应）

MotionEvent.ACTION\_MOVE 滑动View

MotionEvent.ACTION\_CANCEL 结束事件（非人为原因）

2、事件列：从手指接触屏幕 至 手指离开屏幕，这个过程产生的一系列事件。注：一般情况下，事件列都是以DOWN事件开始、UP事件结束，中间有无数的MOVE事件，如下图：



3、事件在哪些对象之间进行传递？答：Activity、ViewGroup、View。

4、事件传递的顺序：Activity -> ViewGroup -> View。

即：1个点击事件发生后，事件先传到Activity、再传到ViewGroup、最终再传到 View

5、安卓里经常会有多个控件重叠，即ViewGroup包含View的情况。

这个时候点击到子View时，其实也是同时点到ViewGroup这个父控件的，那是把这个点击事件分给Parent呢还是Child呢？

这里我们就要了解下安卓中的TouchEvent事件分发机制啦。

（1）俩ViewGroup和一个View，方法全部默认不修改~

嵌套布局则当点击到Child上时，Touch事件的相关方法调用顺序就是

grandpa dispatchTouchEvent ACTION\_DOWN

grandpa onInterceptTouchEvent ACTION\_DOWN

--- parent dispatchTouchEvent ACTION\_DOWN

--- parent onInterceptTouchEvent ACTION\_DOWN

--- --- child dispatchTouchEvent ACTION\_DOWN

--- --- child onTouchEvent ACTION\_DOWN

--- parent onTouchEvent ACTION\_DOWN

grandpa onTouchEvent ACTION\_DOWN

（2）核心在于ViewGroup的dispatchTouchEvent方法

为了方便理解，我们缩减下代码，如下方法的作用是将屏幕点击事件向下（子一级）传递到目标控件上，或者传递给自己，如果自己就是目标的话。

如果事件被（自己或者下面某一层的子控件）处理掉了的话，就返回true，否则返回false。

那问题来了，如果我没有child了，或者我就是一个View，那我的dispatchTouchEvent返回值要如何获取呢？

这种情况下就会使用父类的dispatchTouchEvent方法，

也就是调用View类中的实现，简化代码如下

（3）那onInterceptTouchEvent拦截方法是做什么用的呢？

该方法用于拦截事件向下分发，

当返回值为true时，就会拦截TouchEvent不再向下传递，直接交给自己的onTouchEvent方法处理；返回false则不拦截。

（4）最后还剩个onTouchEvent方法，方法的主体内容其实是处理具体操作逻辑的，是产生一次点击还是一次横纵向的滑动等。

而他的返回值才会影响整个事件分发机制，其意义在于通知父级的ViewGroup们是否已经消费找到目标Target了。

